Contents

PART 1

Introduction

Super Apple™ BASIC is a stepping-stone in sophistication toward advanced languages such as Pascal that do not require the user to throw away his understanding of BASIC or undergo complete retraining in a new language. It also enhances the performance of the Apple II integer and Applesoft BASIC languages by using the entry format of the new advanced languages while retaining the familiar BASIC functions. The user of this program thinks in BASIC but creates programs in a new meta-language with many improvements over BASIC.

The purposes of Super Apple™ BASIC are:

1. To provide a sophisticated and powerful text editor entry format for the creation of understandable computer programs.
2. To eliminate BASIC language characteristics (i.e. line numbers, cryptically short variable names, lack of editing capability) that reduce the user's ability to understand and improve his programs.
3. To provide a compiled form for the program that uses minimum memory and runs optimally fast.

An advanced 80-column text editor is used to insert or delete characters, words, lines, or subprograms anywhere in a program text file. Frequently-used subroutines may be written and saved to disk, then inserted in developing programs as needed.

Because of these features, a library of modular program routines may be created and used to put together complex programs very quickly. Here is an example of such a program:

```
                [ DISPLAY MULTIPLICATION MATRIX ]
@MATRIX     FOR @VERTICAL = 1 TO 8
              FOR @HORIZONTAL = 1 TO 8
                  @RESULT = @VERTICAL * @HORIZONTAL
                VTAB @VERTICAL
                HTAB @HORIZONTAL * 4
                PRINT @RESULT;
              NEXT @HORIZONTAL
            NEXT @VERTICAL
          END
```

BAD NEST

```
FOR V = 1 TO 8
FOR H = 1 TO 8
R = V * H
VTAB V
HTAB H * 4
: R
NEXT H
NEXT V
```

*Apple is a trademark of Apple Computer Company, Inc. and is not affiliated with Hayden Book Company, Inc.

The Super Apple™ BASIC compiler takes this program and converts it into a maximally compact format for use:

```
40 FOR A = 1 TO 8 : FOR B = 1 TO 8 :n C = A * B :
   VTAB A : HTAB B * 4 :
      PRINT C ; : NEXT B : NEXT A : END
```

If this routine were to be called from another part of a larger program, one would type GOTO @MATRIX. There is no need to worry about line numbers because there aren't any.

PART 2

Use of the Text Editor

STARTING UP

To begin, place the program diskette in your disk drive and turn on the Apple II. The following menu will appear:

```
**********************************
***                            ***
***          SUPER APPLE BASIC     ***
***          BY PAUL LUTUS         ***
***    COPYRIGHT © 1980 HAYDEN BOOK CO.  ***
***          ALL RIGHTS RESERVED   ***
***                            ***
**********************************

OPTIONS –
[ E ]  EDIT      FILE
[ C ]  COMPILE FILE
[ P ]  PRINT     FILE
[ D ]  ACCESS DOS
[ Q ]  QUIT
CHOOSE [ LETTER ] :
```

In normal program use, one uses the [**E**] EDIT option to create a program, then the [**C**] COMPILE option is used. For this example, press **E**. The editor "prompt" line will then appear at the top of the screen:

```
ESC)CURS A)LFT/RT D)OS S)TRNG B)EG E)ND
```

The prompt line will always appear at the top of the screen while the editor is in use. It serves to remind the user of some of the more frequently-used editor options. The letters to the left of the parentheses [example: D)OS] are ConTRoL characters.

Editor ConTRoL Characters

Commands are made to the text editor through the use of ConTRoL characters. To type a ConTRoL character, press the **ConTRoL** key (at keyboard left), and, while holding it down, press the letter key for the ConTRoL function.

CTRL A
80 column

ConTRoL A: Screen Side Control

Right now, you are looking at the *left* 40 columns of an 80-column display. Press **ConTRoL A** to see the *right* hand side of the display (and prompt line):

```
L)INE W)ORD K)EEP I)NSRT Z)FOLLOW Q)UIT
```

Press **ConTRoL A** again to bring the cursor (flashing square) back into view. The Apple II screen can display 40 characters horizontally by 24 characters vertically. This screen width is actually a "window" looking into the 80-column Super Apple™ BASIC text editor display format. The window can be adjusted to "look" at any part of the file.

EDITOR ENTRY FORMAT

Erasing Memory

CTRL shift P clear memory

To clear the screen, and the text file memory, press **ConTRoL/SHIFT/P**. This means to press the **ConTRoL** key, the **SHIFT** key, and the **P** key at the same time. This function requires the pressing of three keys because it erases memory, something you won't want to happen often. A prompt will appear:

```
SHIFT P) – ERASE MEMORY (Y/N) ?
```

This is further protection against accidental memory erasure. Having responded with **Y** you will be presented with a clear screen and file. Now try typing something:

```
NOW IS THE TIME FOR ALL GOOD MEN TO COME TO THE AID OF THEIR COUNTRY.
```

spacebar = tab 12 if at col. 0

You will notice that when you first press the **SPACE BAR** the cursor jumps over to screen column 12 (leftmost column = 0). This is part of a label entry format to be explained later. For now, begin each entry with a space so that the entries will not be broken up. You will further notice that an automatic return to column 12 will occur at some point in the typing. This will be made some time after column 72, in such a way that words won't be split.

ConTRoL Z: Follow the Cursor

CTRL Z auto cursor

As you type, the cursor will move to the right and will eventually disappear off the right edge of the screen. You may press **ConTRoL A** again to bring it back into view, or you may press **ConTRoL Z**.

ConTRoL Z makes the display "follow" the cursor, no matter where it goes. If you don't see the cursor on the screen, and need to find it, press **ConTRoL Z**.

FILE ConTRoL CHARACTERS

Most ConTRoL character references in these instructions refer to direct editor commands. This section explains how to place ConTRoL characters in your *program*, ConTRoL D for example, so that the program may gain access to the Disk Operating System, or DOS.

ESC allows a control

Here's how to insert a ConTRoL character in your program: Press the **ESCape** key (keyboard upper left) *once*. The screen cursor is a flashing C (C=ConTRoL).

Now press the letter of the desired ConTRoL character. The ConTRoL character will appear on the screen as an inverse (black on white) character. All inverse characters on the screen will be ConTRoL characters in your program. The **ESCape** key must be pressed once for each ConTRoL character entered.

Character Deletion

If you make a mistake, you may press the **LEFT ARROW** key to delete it. But suppose you want to change the word MEN to PEOPLE (in the previous example) in order not to

win the enmity of Gloria Steinem. You are already past that word. How shall you change only the word MEN without disturbing the other words?

Cursor Control

Here's how: Press the **ESCape** key *twice*. The cursur now has a + on it. This means that the *Cursor Control Mode* has been enabled.

Cursor Control Mode means that, by using the letter keys **I**, **J**, **K**, **M** (I = up, J = left, K = right, M = down) you may move the cursor anywhere on the screen without disturbing the text. Once you have moved the cursor to where you want to insert or delete characters, just begin typing again. Now we change the word MEN to PERSONS (the cursor is marked ☐):

Replacing a Word

1. Press **ESCape** (keyboard upper left) *twice.* The cursor will be +
2. Move the cursor to the desired place, using the cursor control keys **I**, **J**, **K**, **M**:

   NOW IS THE TIME FOR ALL GOOD MEN ⊞ TO COME TO THE AID OF THEIR COUNTRY.

3. Delete the word MEN with the **LEFT ARROW** key:

   NOW IS THE TIME FOR ALL GOOD ☐ TO COME TO THE AID OF THEIR COUNTRY.

4. Type PERSONS:

   NOW IS THE TIME FOR ALL GOOD PERSONS ☐ TO COME TO THE AID OF THEIR COUNTRY.

Other Deletion Options

You may also delete characters a word at a time by pressing **ConTRoL W**, or a line at a time by pressing **ConTRoL L**. Try it!

Character Recovery

All characters deleted using these features may be gotten back by pressing the **RIGHT ARROW** key. This means that you may change the order of words or lines by deleting at the original position, moving the cursor, then recovering the characters using the **RIGHT ARROW** key. Try this also.

If you delete more than 256 characters without recovering any, the oldest deletions will begin to be lost. The most recent 256 deleted characters are saved for you.

Vertical Display Scrolling

If you type a great deal, the text will scroll upward so that some text will be above the top of the screen. To get back to that text, simply move the cursor to the top of the screen. The text will scroll downward to keep the cursor on the screen, all the way to the beginning of the file if necessary.

To move up 12 lines at a time, press **ConTRoL R**. To move down 12 lines at a time, press **ConTRoL C**. To jump to the file beginning, press **ConTRoL B**. To jump to the file end, press **ConTRoL E**.

ConTRoL S: String Search

Let us say that you have a long file, and want to find something without scrolling completely through the file. To do this, press **ConTRoL S** (S=String, or a group of characters). The following prompt will appear:

```
S) – ENTER /NOW/TO BE/ :
```

If you just want to find something, and don't need to replace it, simply enter the thing to be sought: /FIND THIS/. If you want to replace something with something else, enter both: /FIND THIS/REPLACE WITH THIS/.

The / separates the search string from the replacement. Any character may be used instead of / , but the character used may not be part of either entry.

After you have made your entry, the program will search *forward* through the file for the search string. This means that you must be at the beginning of the file to catch every occurrence of the string. Another prompt appears while the search is underway:

```
S) – R=REPLACE,.RETURN=NEXT, SPACE=QUIT
```

This means that you should press the **RETURN** key to continue the search, press **R** to replace the displayed string with the specified replacement, or simply press the **SPACE BAR** (or any key except **RETURN** or **R**) to exit the string function. Example:

You have typed:

```
SACRAMENTO IS THE CAPITAL OF CALIFORNIA
```

After another 10,000 lines, you remember that you have misspelled CAPITOL. To find it. you:

1. Jump to the file beginning, using **ConTRoL B**.
2. Press **ConTRoL S**, and enter: /CAPITAL/CAPITOL/
3. The program will find the error and stop. It is waiting for you to decide to replace, go on, or quit. If you press **R**, the replacement will be made, and the search will continue. At the end of the search, the cursor will be positioned at the end of the file.

NOTE: If you press the **RETURN** key without any entry for the string search feature, the previous entry (if there is one) will be used for the search.

ConTRoL D: DOS Commands

Disk Operating System commands may be made by pressing **ConTRoL D**. A prompt will appear:

```
D = DISK L)OAD S)AVE (FILENAME)
(OTHER DOS COMMANDS) Q)UIT:
```

If you want to save the contents of a file named "TEST" to disk, you type S TEST . To load such a file, type L TEST . If you want to specify a disk drive or slot other than the last used one, add this information after the file name: L TEST,S6,D1 . Other DOS commands (CATALOG, DELETE, etc.) may be entered.

The file is saved with the prefix BASIC. automatically added to the file name. This is to identify the file with the Super Apple™ BASIC system. There is no need to type this prefix; it is added automatically. To quit this feature and return to the editor, type **Q**.

ConTRoL K: Keep a File Segment

To save *part* of a file, hit **ConTRoL K**. A prompt will appear at screen bottom:

```
K – SAVE FILE SEGMENT
ENTER MARKER:
```

The "marker" to be entered is the character string at the beginning of the desired file segment. For example, suppose you had typed in a lot of text, and wanted to save only one sentence:

```
MILITARY MUSIC IS TO MUSIC AS MILITARY JUSTICE IS TO JUSTICE.
```

To save this sentence, you would move the cursor to the right of the period, then press **ConTRoL K** and enter MILITARY MUSIC as the marker. The ConTRoL K feature then will search backward from the cursor position for the marker. The marker does not have to be on the screen, and the segment can be any length. If the program fails to find the marker, a bell will ring and the ConTRoL K feature will be exited.

If the marker *is* found, a file name will be asked for, under which the file will be saved. The file that is saved will contain all the characters between the marker (including the marker) and the cursor. This file may then be used as a file by itself, or inserted (see ConTRoL I below) somewhere else in this or another file.

The selection of a marker is very important. It must be unique between the cursor and the desired segment beginning. In the example above, if MILITARY had been entered as the marker, only the last 5 words would have been saved. Sometimes it is necessary to insert a special marker at the segment beginning (such as <<>>) to be sure that it is unique. This special marker may be removed later.

ConTRoL I: Insert a File or File Segment

ConTRoL I will insert a disk file at the present cursor position. Simply place the cursor where you want to file to go and press **ConTRoL I**. A prompt will appear:

```
I) – INSERT FILE
ENTER FILE NAME:
```

The disk file will be merged with the file in memory at the cursor position. This feature, and the previously described ConTRoL K feature, are useful for creating and using a library of useful subroutines. These subroutines may then be called up as needed in a developing program. This makes it possible to put together large programs very quickly, using a library of standard routines proven in earlier programs, which are inserted as needed.

ConTRoL F: Free Memory

As you enter program information, memory locations are used up. To find out how many characters of memory are left, press **ConTRoL F**. A display will appear at screen bottom:

```
FREE MEMORY = [characters of remaining memory]
```

It is a good idea to save a file before it completely fills available memory, so that editorial changes may be added in the future without memory overflow.

ConTRoL Q: Quit

To leave the text editor and return to the main menu, press **ConTRoL Q**. A prompt will appear:

```
Q) – UPDATE WORKFILE (Y/N) ?
```

This means that a disk file named WORKFILE will be given the present memory contents. This workfile is always loaded into text file memory when Super Apple™ BASIC is run. It is a "scratchpad" text file that is intended to contain the most recent work. After a program file is completed it should be saved using some other name with ConTRoL D – DOS. Then the workfile may be used for something else.

PART 3

Super Apple™ BASIC Programming

The preceding section described the text editor used for program entry. This section explains the Super Apple™ BASIC language and features, and the entry method for creating programs from them. It is assumed that the reader is familiar with either integer or Applesoft BASIC.

SUPER APPLE™ BASIC STRUCTURE

The structure can be divided into two main categories: *labels* and *BASIC operators*. BASIC operators (FOR, NEXT, IF, THEN, etc.) are those expressions that have meaning in an ordinary BASIC program. Labels are used to (1) identify arithmetic or string variables, and (2) identify locations in the program. These two kinds of labels are called "variable" and "location" labels respectively.

A label always starts with @ (**SHIFT P**). A label must contain only the letters A through Z. Up to ten characters may be used for location labels. There is no limit to the number of characters in a variable label. There may not be embedded spaces in a label. Here is an example of the use of labels:

```
@POLAR      @TERMX = @XVALUE ^ 2
            @TERMY = @YVALUE ^ 2
            @MAGNITUDE = SQR (@TERMX + @TERMY)
            @ANGLE = ATN (@YVALUE / @XVALUE)
            RETURN
```

The label @POLAR is a location label. Elsewhere in the program might be a line saying GOSUB @POLAR. The other labels are variable labels.

Location labels begin at the left edge of the screen. If you are entering one, simply type it in, starting after a carriage return. If you are entering a variable, always press the **SPACE BAR** first if you are at the left edge of the screen.

There must be a location label before the first BASIC operator in the program for correct compilation. If the following example were the first file entry, it would *not* compile correctly:

```
            CALL @HOME
@START      FOR @VALUE = 0 TO 30
```

With the just-noted exception, only use location labels at places in the program that will be accessed from elsewhere (there is another use for location labels to be discussed later with respect to Applesoft). Each location label must be unique in the program. For example, there can be only one @POLAR at the left edge of the screen. This label may be called GOSUB @POLAR any number of times, of course.

The colon is not used in Super Apple™ BASIC. Each statement is followed by a carriage return instead. Example:

```
                [ DISPLAY MULTIPLICATION MATRIX ]
@MATRIX         FOR @VERTICAL = 1 TO 8
                   FOR @HORIZONTAL = 1 TO 8
                      @RESULT = @VERTICAL * @HORIZONTAL
                      VTAB @VERTICAL
                      HTAB @HORIZONTAL * 4
                      PRINT @RESULT;
                   NEXT @HORIZONTAL
                NEXT @VERTICAL
               END
```

The indentations shown are not necessary for proper compilation, but they greatly improve the readability of the program and are strongly recommended. They are used to indicate the structure of the program, with the innermost routines indented the farthest.

PROGRAM COMMENTS

Comments that are not to appear in the final BASIC program are placed inside brackets [ ] . The left bracket is gotten by pressing **ConTRoL** N. The right bracket is gotten by pressing **SHIFT M**.

An example of two kinds of comments:

This comment will appear only in the Super Apple™ BASIC text file:

```
[ DISPLAY MULTIPLICATION MATRIX ]
```

This comment will appear there, as well as in the compiled BASIC program:

```
REM DISPLAY MULTIPLICATION MATRIX
```

The second (REM) comment type must be followed by a new location label on the next line. This drawback, and the fact that the (REM) comment uses memory in the compiled version, means that it should be used sparingly.

The comments in brackets may appear on lines by themselves, or at the end of entered lines:

```
@VALUE = @VALUE + 1 [ INCREMENT ]
```

A bracketed comment may not immediately follow a location label. The following entry will cause *incorrect* compilation:

```
@RESULT         [ ADD XVALUE AND YVALUE ]
                @TOTAL = @XVALUE + @YVALUE
                RETURN
```

There is a disk file called TREK that is intended for compilation as an integer BASIC program that shows these techniques. Loading and reading the text file will give a feeling for what Super Apple™ BASIC looks like. It is also a good idea to see the compiled version (also TREK) for comparison.

APPLESOFT IF-THEN CASE

In Applesoft BASIC (unlike integer BASIC), if a comparison test fails, the line is exited and subsequent statements are not executed. This carries over to Super Apple™ BASIC in the following way:

```
@TEST       IF @FLAG THEN @XVALUE = 0
            @YVALUE = 0
            @ZVALUE = 0
```

If @FLAG is true, all the statements will be executed. If @FLAG is false, *none* of the statements will be carried out. To avoid this problem, group statements with the tests for which they are appropriate and use a new location label to segregate tests. Example:

```
@TESTA      IF @FLAGA THEN (statement for A)
            (other statements for A)
@TESTB      IF @FLAGB THEN (statement for B)
            (other statements for B)
```

BASIC TO SUPER APPLE™ BASIC CONVERSION

You may want to transfer a BASIC program into the Super Apple™ BASIC text editor for revising into the appropriate format and other improvements. Here is how:

1. Let us say you have a BASIC program with line numbers between 10 and 2000. Add the following routine to the program:

```
10000 D$ = " "  (CONTROL D)
10010 PRINT D$ ; "OPEN TRF"
10020 PRINT D$ ; "WRITE TRF"
10030 LIST 0, 9999
10040 PRINT D$ ; "CLOSE"
10050 END
```

2. Now type RUN 10000 . A file called TRF contains the program that will be saved to disk.
3. Run Super Apple™ BASIC, enter the editor, press **ConTRoL D**, and type EXEC TRF .
4. The BASIC program will be entered into the text editor.

After this procedure, it will be necessary to substantially revise the program to make it acceptable to Super Apple™ BASIC. All line numbers called from elsewhere in the program with GOTO and GOSUB should be replaced with descriptive labels. Other line numbers (not called) should be eliminated. All variable names should be replaced with Super Apple™ BASIC labels that describe their functions.

PART 4

Compiling the Program

After the text file has been created it may be compiled into a form suitable for running in integer or Applesoft BASIC. To compile the program, exit the text editor and select option [**C**] COMPILE. A prompt will appear:

```
BASIC: (I)NTEGER OR (A)PPLESOFT ?
```

Enter **I** if the program is to be run in integer BASIC. Enter **A** if the program is to be run in Applesoft. The reason for this entry is that Applesoft BASIC will accept a much longer line entry than integer BASIC. The lines compiled by Super Apple™ BASIC are therefore longer and make more efficient use of memory in Applesoft.

After you have made this entry, a label will appear:

```
[ COMPILING ]
```

While this label is visible a disk file is being created, so don't press the **RESET** key or touch the disk drive.

HOW THE COMPILER WORKS

The first step in compilation is to replace all variable labels with BASIC-style labels that are as short as possible. The replacement begins with the letters A through Z, then A0 through A9, B0 through B9, and so forth. The entire file is scanned for cases of the label and the replacement is made to each case.

If a label is found in the left columns, it is assumed to be a location rather than a variable and it is passed up until later.

The next step begins compilation. The first location label that is found is given the line number 40. Location labels are given line numbers in increments of 40 (40, 80, 120, etc.). Statements between location labels are added to the developing line using colons. If the maximum line length for the BASIC used is exceeded, intermediate line numbers are given and the process continues.

The intermediate line numbers are given the following numeric value: the last location line number plus (2, 4, 6, 8, etc.). This means that there can be 20 intermediate line numbers between location labels, or 4700 BASIC characters between specified locations. Therefore, for all practical purposes, the user need not concern himself with the distance between location labels.

COMPILATION ERRORS

If there is a variable label that has accidentally been used for a location label, the variable name will be replaced by the line number for that location. Program run-time errors are frequently traceable to this cause.

HINT: To find such errors, enter each label in the editor ConTRoL S String feature, and scan the file to verify that the label is being used consistently.

If the user has been writing BASIC programs for some time, there may be irresistible impulses to: use the colon, enter a line number, use a variable name without the special Super Apple™ BASIC token @, or a number of other very bad habits. These aberrant behaviors go away as time and experience are gained.

LOADING AND RUNNING THE BASIC PROGRAM

The compiled BASIC program is contained in a disk file called TRANSFER. This file is moved to BASIC by way of the DOS EXEC feature. To load the program into BASIC, type Q (quit) from the main menu.

Now verify that the BASIC is appropriate to the program (if you had selected (A)PPLESOFT at the compiler, there must be a] to the left of the cursor). Now type EXEC TRANSFER. The disk file will be transferred to BASIC. If you want to see the lines as they are entered (a good diagnostic method), type: MON I, O, C then EXEC TRANSFER. The BASIC lines will be visible as they are entered.

PROGRAM SPEED IMPROVEMENTS

There are ways to further increase the speed of the BASIC program beyond the results of Super Apple™ BASIC. One of these is to move all frequently used routines to the beginning of the program. This is most easily done using the described text editor ConTRoL K and ConTRoL I file segment features.

Another speed improvement is to declare frequently used variable names at the start of the program. This assures that they will be placed in the lowest BASIC variable table locations (meaning that they will be found quickly in the running program). This also means that they will be assigned one of the 26 single-letter variable names by the compiler, another speed advantage.

PARTITIONING LONG PROGRAMS

In some cases it will be necessary to compile large programs in sections. This is done in the following way:

1. Create and compile each program segment, load it into BASIC, and save it to disk.
2. After all segments have been created, renumber the segments using the available renumber programs (integer: utility ROM. Applesoft: DOS 3.2 Master Diskette Renumber program) so that they have unique line numbers.
3. Create a master menu program to control all the segments. The menu program must include numeric branches that specify the starting line numbers of each segment.
4. Finally, merge all the segments using the merge feature of the Applesoft renumber program, or the Append feature of the utility ROM for integer BASIC programs.

PART 5

Printing the Program Text File

Super Apple™ BASIC has an intrinsic printing and formatting program that will operate a hard-copy printer through the Apple II peripheral slots. There is also a built-in serial interface that operates through the game I/O at 1200 baud (see Appendix B).

To print the text file, connect the printer interface card (with power turned *off*) to the desired peripheral slot and prepare the printer for use. Run Super Apple™ BASIC and load the text file (if the text file to be printed is the workfile, it will be loaded automatically).

Select main menu option [**P**] PRINT. A prompt will appear:

```
PRINTING CONSTANTS –
(A) PRINTER ADDRESS     =
(B) PAGE LENGTH         =
(C) LEFT MARGIN         =
(D) TOP MARGIN          =
(E) BOTTOM MARGIN       =
(F) PRINT "@"           =

(P) BEGIN PRINT

ENTER (LETTER) :
```

(A) PRINTER ADDRESS: This address is gotten from the user's manual for the printer interface card being used. Special addresses may be entered if desired. If **S** is entered, the built-in serial interface will be enabled (see Appendix B).

(B) PAGE LENGTH: This is the number of lines between the top of one page and the top of the next. It is usually 66 lines.

(C) LEFT MARGIN: This is the distance in columns between the left edge of the paper and the beginning of printing. The right margin depends on the text entries in the file, but is rarely greater than 72 columns.

(D) TOP MARGIN: This is the distance in lines between the page title and number and the first printed line.

(E) BOTTOM MARGIN: This is the distance in lines between the last printed line of a page and the title and number of the next page.

(F) PRINT @: This option prevents the printing of the special Super Apple™ BASIC token @. Although essential for program formatting, the token doesn't add to the readability of the listing and can therefore be suppressed. If this symbol appears in the program in a print statement (between quotes), it will be printed regardless of the setting of this option.

(P) BEGIN PRINT: Select **P** after all the printing constants are satisfactory.

When you have selected **P** another prompt will appear:

```
ENTER PAGE HEADING
(NO ENTRY = QUIT) :
```

If you press the **RETURN** key without making an entry, the printing option will be exited. If you don't want a heading, simply press the **SPACE BAR** once, then **RETURN**. The heading and a page number will be centered at the top of each printed page. After the heading has been entered, printing will commence. ConTRoL characters that have been inserted in the program (see File ConTRoL characters in Part 2) will be printed as lower-case letters; all other characters will be printed as upper-case.

APPENDIX A

Short-Form Listing of Editor ConTRoLs and Commands

ConTRoL A = Switch screen sides
Z = Follow cursor
D = Access Disk Operating System
B = Jump to file beginning
E = Jump to file end
S = String search and replace
R = Move up 12 lines
C = Move down 12 lines
F = Print free memory
K = Keep file segment to disk
I = Insert Disk file at cursor
L.ARROW = Delete one character
W = Delete a word (preceded by a space)
L = Delete a line (preceded by a carriage return)
R.ARROW = Recover last deleted character
SHIFT P = Erase memory
Q = Quit editor

ESCape (once) = Set ConTRoL character entry mode

(cursor = C)
ConTRoL characters appear inverse (black on white).

ESCape (twice) = Set cursor control mode:

(cursor = +)

I Move up 1 line
J Move left 1 character
K Move right 1 character
M Move down 1 line
(space) Return to normal mode

APPENDIX B

Built-In Serial Interface

NOTE: The following is a technical discussion of the built-in serial interface. It has no direct bearing on the operation of the Super Apple™ BASIC system.

Entering **S** instead of a printer address during entry of printer constants enables a software-implemented serial interface. The connection point for this interface is the Apple II game I/O socket. To make connection it is necessary to acquire a 16-pin I.C. plug (such as that used on the game controls).

A 1200 baud rate is available. Electrical interface is normal 5-volt TTL. A standard serial signal with one sync and one stop bit is generated and appears at game I/O pin 15. The printer's read ready signal is connected to pin 2. Ground connectors are made to pin 8.

Because these interfaces are not buffered, great care must be taken in making the connections. Don't connect or disconnect the printer while either it or the Apple II are turned on. Make sure that the printer will electrically interface with the Apple II.